AAR-based decomposition method for lower bound limit analysis

J. J. Muñoz, **N. Rabiei**
Department of Applied Mathematics III
Laboratory of Numerical Analysis (LaCàN)
Universitat Politècnica de Catalunya (UPC), Urgell 187, 08036 Barcelona, Spain

Abstract

Despite recent progress in the optimisation techniques, finite element stability analysis of realistic three-dimensional (3D) problems is still hampered by the size of the resulting optimisation problem. Current solvers may take a prohibitive computational time, if they give a solution at all. Possible remedies to this are the design of adaptive de-remeshing techniques, decomposition of the system of equations, or the decomposition of the optimisation problem. In this paper we concentrate on the last approach, and present an algorithm especially suited for limit analysis.

The optimisation problems in limit analysis are convex but non-linear. This fact renders the design of decomposition techniques specially challenging. The efficiency of general approaches such as Benders or Dantzig-Wolfe is not always satisfactory, and strongly depends on the structure of the optimisation problem. We here present a new method that is based on rewriting the feasibility region of the global optimisation problem as the intersection of two subsets. By resorting to the Averaged Alternate Reflections (AAR) in order to find the distance between the sets, we achieve to solve the optimisation problem in a decomposed manner. We illustrate the method with some representative problems, and comment its efficiency with respect to other well-known decomposition algorithms.

1 Introduction

Computational limit analysis aims to accurately compute the bearing capacity of structures. Mathematically, this can be stated as numerically solving the following maximisation (static) problem:

$$\begin{aligned} \lambda_{opt} = \max_{\lambda,\boldsymbol{\sigma}} & \lambda \\ s.t. & \nabla\cdot\boldsymbol{\sigma} + \lambda\boldsymbol{f} = \boldsymbol{0}, && \forall \boldsymbol{x}\in\Omega \\ & \boldsymbol{\sigma}\boldsymbol{n} = \lambda\boldsymbol{g}, && \forall \boldsymbol{x}\in\Gamma_n \\ & [\![\boldsymbol{\sigma}\boldsymbol{n}]\!] = \boldsymbol{0}, && \forall \boldsymbol{x}\in\Gamma_i \\ & \boldsymbol{\sigma}\in\mathcal{B} \end{aligned} \tag{1}$$

Here, Ω is the domain of the body, while $\boldsymbol{\sigma}$, $\boldsymbol{f}$ and $\boldsymbol{g}$ are respectively the stress tensor, the volumetric loads, and the boundary loads. The conditions

in the optimisation problem (1) correspond to the equilibrium conditions of a domain Ω, with applied boundary loads $\boldsymbol{g}$ on the boundary $\Gamma_n \subseteq \partial\Omega$, and with some potential discontinuities Γ_i. The set $\mathcal{B}$ represents the admissible domain for the plasticity criteria of the material.

Different discretisations of the optimisation problem in (1) yield different static and kinematic formulations that give respectively lower [LS02, LSKH05] or upper bounds [KLS07] of the exact optimal load factor λ_{opt}, and the two type of solutions may be in turn combined for designing remeshing strategies [MBHP09]. The method has been well studied and applied for instance in the analysis of anchors [MS10, MLH13], masonry structures [GSP10] or inhomogeneous materials [Bd14]. We will here focus on the lower bound optimisation problem, although the ideas described below can be also applied to other formulations. By using a piecewise linear finite element discretissation of the stress variable $\boldsymbol{\sigma}$, and after using a linear transformation of the stresses, the analytical problem in (1) can be turned into the following finite optimisation problem [LS02, MBHP09]:

$$\begin{aligned} \lambda^* = \max_{\lambda,\mathbf{x}} & \lambda \\ s.t. & \mathbf{A}\mathbf{x} + \lambda\mathbf{f} = \mathbf{b} \\ & \mathbf{x} \in \mathcal{K} \end{aligned} \tag{2}$$

The vector $\mathbf{x}$ includes all the nodal components of the stress-like variable $\mathbf{x}$, which is a linear transformation of the stresses $\boldsymbol{\sigma}$, in such a manner that the new admissible set $\mathcal{K}$ is formed by the products of second order cones (SOCs), that is $\mathcal{K} = \mathcal{K}_1 \times \ldots \mathcal{K}_N$, with $\mathcal{K}_i = \{\mathbf{y} \in \mathbb{R}^n | y_1 \geq \sqrt{y_2^2 + \ldots + y_n^2}\}$. The membership constraint $\mathbf{x} \in \mathcal{K}$ can be then easily dealt with by using standard optimisation software such as SDPT3 [TTT06], Mosek [MOS05] or Sonic [Lya04]. The static formulations solved here are such that the optimum value of the optimisation problem in (2), denoted by λ^*, is a lower bound of the exact solution in (1), i.e. $\lambda^* \leq \lambda_{opt}$.

Due to the size of the resulting optimisation problem in (2), limit analyses on three dimensional domains are scarce. Despite recent progress in the optimisation solvers, it is still very much desirable to design new methods that allow to reduce the cost of the solution process. One of the possible remedies is to decompose the global problem in (2) into smaller sub-problems, which can be solved at a much lower cost. This idea is not new, and has already been used by [CT94, Kan83] using proximal-point decomposition, and by [PLP09] and [KPSP10] using overlapping domains. We note that no optimal decomposition strategy exists in the optimisation literature, and that the standard techniques such as Benders [Ben62, Geo72], Dantzig-Wolfe [CCMGB06, DW60], or primal and dual decomposition [BXMM07] require an exceeding number of iterations for the non-linear problem in (2) [MRLH13].

The proposed algorithm is based on computing the distance between two feasilibty sets, which we find by using the method of Averaged Alternating Reflections (AAR). We present first in Section 2.1 this method, which we will

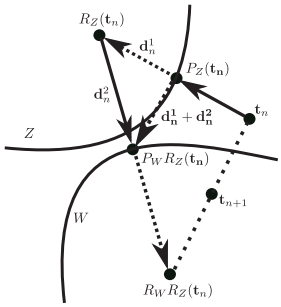

Figure 1: Illustrtion of the AAR algorithm for finding the distance between the two sets Z and W.

relate to our decomposition algorithm in Section 2.2. Section 2.3 describes the method, Section 3 applies it to some representative examples, and Section 4 discusses the main results.

2 AAR-based decomposition algorithm

2.1 AAR algorithm

The AAR algorithm consists on finding the distance between two sets Z and W [BC10]. It is clear that if the distance between the sets W and Z, denoted by $d(W, Z)$, is equal to zero, and the sets are compact, there is a common element $\bar{\mathbf{t}} \in Z \cap W$. The AAR algorithm searches for such element $\bar{\mathbf{t}}$ by computing the fix points of the following iterative process:

$$\mathbf{t}_{n+1} = \mathbf{T}(\mathbf{t}_n), \quad \text{with} \quad \mathbf{T} = \frac{\mathbf{R}_W \mathbf{R}_Z + \mathbf{I}}{2}. \tag{3}$$

The transformations $\mathbf{R}_W = 2\mathbf{P}_W - \mathbf{I}$ and $\mathbf{R}_Z = 2\mathbf{P}_Z - \mathbf{I}$ are the reflections on the sets W and Z, with $\mathbf{P}_W$ and $\mathbf{P}_Z$ the projections onto the same sets, respectively. Figure 1 illustrates the meaning of the transformation $\mathbf{T}, \mathbf{R}$ and $\mathbf{P}$. It is demonstrated in [BC10], that when $d(W, Z) = 0$, the iterative process in (3) converges towards a fix point such that $\bar{\mathbf{t}} = \mathbf{T}(\bar{\mathbf{t}})$ and $\bar{\mathbf{t}} \in Z \cap W$. And conversely, when $d(W, Z) > 0$, the algorithm diverges, giving a series of increasing values $\alpha_n = ||\mathbf{t}_{n+1} - \mathbf{t}_n||$.

2.2 Decomposition of lower bound optimisation problem

We propose here a decomposition method which splits the domain Ω into two non-overlapping domains, Ω_1 and Ω_2, with $\Omega = \Omega_1 \cup \Omega_2$. The global optimisation

problem in (2) is also rewritten into the following partitioned form:

$$\begin{aligned}\lambda^* = \max_{\lambda,\mathbf{x}_1,\mathbf{x}_2}\ & \lambda \\ s.t. & \mathbf{A}_1\mathbf{x}_1 + \lambda\mathbf{f}_1 = \mathbf{b}_1 \\ & \mathbf{A}_2\mathbf{x}_2 + \lambda\mathbf{f}_2 = \mathbf{b}_2 \\ & \mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2 = \mathbf{0} \\ & \mathbf{x}_1 \in \mathcal{K}_1, \mathbf{x}_2 \in \mathcal{K}_2\end{aligned}$$

where $\mathbf{x}_1 \in \Omega_1$ and $\mathbf{x}_2 \in \Omega_2$ are the stress-like nodal variables in each domain. It will be convenient to rewrite the complicating constraint $\mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2 = \mathbf{0}$ into two constraints, and use a new complicating variable $\mathbf{t}$ in such a manner that the optimisation problem above now reads

$$\begin{aligned}\lambda^* = \max_{\lambda,\mathbf{x}_1,\mathbf{x}_2,\mathbf{t}}\ & \lambda \\ s.t. & \mathbf{A}_1\mathbf{x}_1 + \lambda\mathbf{f}_1 = \mathbf{b}_1 \\ & \mathbf{A}_2\mathbf{x}_2 + \lambda\mathbf{f}_2 = \mathbf{b}_2 \\ & \mathbf{B}_1\mathbf{x}_1 = \mathbf{t} \\ & \mathbf{B}_2\mathbf{x}_2 = -\mathbf{t} \\ & \mathbf{x}_1 \in \mathcal{K}_1, \mathbf{x}_2 \in \mathcal{K}_2\end{aligned} \tag{4}$$

The new variable $\mathbf{t}$ corresponds to the nodal tractions at the common boundary $\Omega_1 \cap \Omega_2$, as illustrated in Figure 2. This variable allows us to decompose in turn the optimisation problem in (4) into the following master problem

$$\begin{aligned}\lambda^* = \max_{\lambda}\ & \lambda \\ s.t. \quad & d(Z(\lambda), W(\lambda)) = 0\end{aligned} \tag{5}$$

where $Z(\lambda)$ and $W(\lambda)$ are the following feasible sets:

$$\begin{aligned}W(\lambda) &= \{\mathbf{t} | \mathbf{A}_1\mathbf{x}_1 + \lambda\mathbf{f}_1 = \mathbf{b}_1, \mathbf{B}_1\mathbf{x}_1 = \mathbf{t}, \mathbf{x}_1 \in \mathcal{K}_1\} \\ Z(\lambda) &= \{\mathbf{t} | \mathbf{A}_2\mathbf{x}_2 + \lambda\mathbf{f}_2 = \mathbf{b}_2, \mathbf{B}_2\mathbf{x}_2 = -\mathbf{t}, \mathbf{x}_2 \in \mathcal{K}_2\}\end{aligned} \tag{6}$$

In mechanical terms, $W(\lambda)$ and $Z(\lambda)$ represent the sets of tractions at the boundary $\Omega_1 \cap \Omega_2$ that are in equilibrium with the given load factor λ, and with admissible stresses $\mathbf{x}_1 \in \mathcal{K}_1$ and $\mathbf{x}_2 \in \mathcal{K}_2$, respectively. If the value of $\lambda = \bar{\lambda}$ is larger than λ^*, that is, when $\bar{\lambda}$ is not globally feasible, then no common traction field $\mathbf{t}$ at the boundary can be found that is in equilibrium with admissible stresses $\mathbf{x}_1$ and $\mathbf{x}_2$, and with the load factor $\bar{\lambda}$. Furthermore, it may occur that at least one of the sets $W(\bar{\lambda})$ or $Z(\bar{\lambda})$ is empty since no equilibrated traction field can be found for the load factor $\bar{\lambda}$. On the other hand, if for a given value $\lambda = \underline{\lambda}$, the intersection $W(\underline{\lambda}) \cap Z(\underline{\lambda})$ is non-empty, then the value $\underline{\lambda}$ is globally feasible, and $\underline{\lambda} \leq \lambda^*$.

Figure 3 shows schematically the sets $W(\lambda)$ and $Z(\lambda)$ on the $(\lambda, \mathbf{t})$-plane, and the situations when $\lambda < \lambda^*$ and $\lambda > \lambda^*$. In each case it can be observed

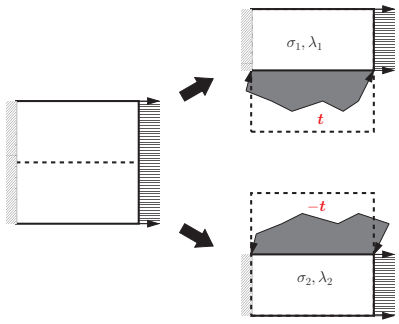

Figure 2: Partitioning of the domain Ω (left) into domains Ω_1 and Ω_2 (right). Variable t are the tractions at the common boundary $\Omega_1 \cap \Omega_2$.

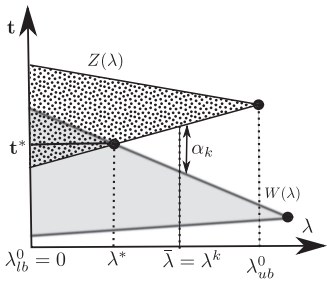

Figure 3: Illustration of the feasibility sets $Z(\lambda)$ and $W(\lambda)$ as a function of λ, and the optimal value λ^*

that:

$$\begin{cases} \lambda \leq \lambda^* \Leftrightarrow W(\lambda) \cap Z(\lambda) \neq \emptyset \Leftrightarrow d(W(\lambda), Z(\lambda)) = 0 \\ \lambda > \lambda^* \Leftrightarrow W(\lambda) \cap Z(\lambda) = \emptyset \Leftrightarrow d(W(\lambda), Z(\lambda)) > 0 \end{cases} \tag{7}$$

These implications justify the form of the optimisation problem given in (5), which in mechanical terms read: find the maximum load factor λ^* such that the two domains Ω_1 and Ω_2 are in equilibrium with a common traction field $\mathbf{t}^*$, and such that the stress variables $\mathbf{x}_1$ and $\mathbf{x}_2$ are plastically admissible and in equilibrium with $\mathbf{t}^*$.

2.3 Master problem and sub-problems

In view of the previous results, the following master problem for solving the optimal problem in (5) is proposed:

- Initialise: find $\lambda^0_{lb} < \lambda^*$ and $\lambda^0_{ub} > \lambda^*$. Set $\lambda^0 = \frac{1}{2}(\lambda^0_{lb} + \lambda^0_{ub})$ and $k = 0$.
- Step 1: Find the distance $\alpha^k = d(W(\lambda^k), Z(\lambda^k))$.
- Step 2:

2.1 If $\alpha^k = 0$: Set $\lambda_{lb}^{k+1} = \lambda^k$ and $\lambda_{ub}^{k+1} = \lambda_{ub}^k$.

2.2 If $\alpha^k > 0$: Set $\lambda_{lb}^{k+1} = \lambda_{lb}^k$ and $\lambda_{ub}^{k+1} = \lambda^k$.

- Step 3: Compute $\epsilon_\lambda = \lambda_{lb}^{k+1} - \lambda_{lb}^{k+1}$.

  3.1 If $\epsilon_\lambda < tol$: STOP.

  3.2 If $\epsilon_\lambda \geq tol$: $\lambda^{k+1} = \left(\lambda_{lb}^{k+1} + \lambda_{ub}^{k+1}\right)/2$, set $k = k+1$ and GO TO Step 1.

Step 1 requires the computation of the distance $\alpha^k = d(W(\lambda^k), Z(\lambda^k))$. This step will be completed by resorting to the AAR algorithm presented in Section 2.1, and will be identified as the sub-problem. In fact, since we are not interested in computing the actual value of the distance, but just in detecting whether α^k is positive, the following sub-problem is proposed:

- Step 1.1. Set $\lambda = \lambda^k$, $\mathbf{t}_0^k = \mathbf{t}^{k-1}$, $n = 0$
- Step 1.2. Solve Sub-problem 1 in (8). Obtain $\mathbf{d}_n^1$ and set $\mathbf{t}_n^k = \mathbf{t}_n^k + 2\mathbf{d}_n^1$.
- Step 1.3. Solve Sub-problem 2 in (9). Obtain $\mathbf{d}_n^2$ and set $\mathbf{t}_n^k = \mathbf{t}_n^k + 2\mathbf{d}_n^2$.
- Step 1.4. Set $\beta_n = ||\mathbf{d}_n^1|| + ||\mathbf{d}_n^2||$ and $\alpha_n = ||\mathbf{d}_n^1 + \mathbf{d}_n^2||$.
  If $\beta_n > \beta_{n+1}$ or $\alpha_n < \epsilon_{1\alpha}$
  - 1.4.1. Set $\alpha^k = 0$. STOP

  elseif $\beta_{n+1} > \beta_n$ and $\Delta\alpha_n < \epsilon_{2\alpha}$
  - 1.4.2. Set $\alpha^k > 0$. STOP

  else
  - 1.4.3. Set $\mathbf{t}_n^k = \frac{\mathbf{t}_{n+1}^k + \mathbf{t}_n^k}{2}$, $n = n+1$, GO TO Step 1.2

Step 1.1. consists on the initialisation of the algorithm. The initial lower and upper bounds, λ_{lb}^0 and λ_{ub}^0, can be computed by setting $\lambda_{lb}^0 = 0$, and $\lambda_{ub}^0 = \min(\lambda_1^0, \lambda_2^0)$, with λ_1^0 and λ_2^0 the optimal values of λ when the global problem is solved with only the constraints of domain Ω_1 and Ω_2, respectively. The sub-problems 1 and 2, in Steps 1.2 and 1.3, are respectively given by

$$\begin{aligned} &\min_{\mathbf{x}_1, \mathbf{d}^1} ||\mathbf{d}^1|| \\ &s.t. \mathbf{A}_1\mathbf{x}_1 + \lambda\mathbf{f}_1 = \mathbf{b}_1 \\ &\quad \mathbf{B}_1\mathbf{x}_1 - \mathbf{d}^1 = \mathbf{t}_n \\ &\quad \mathbf{x}_1 \in \mathcal{K}_1 \end{aligned} \tag{8}$$

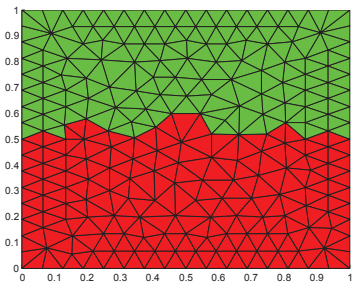

Figure 4: Unstructured mesh for Problem 1 (nelem=402)

and

$$\begin{aligned} &\min_{\mathbf{x}_2,\mathbf{d}^2} ||\mathbf{d}^2|| \\ &s.t. \mathbf{A}_2\mathbf{x}_1 + \lambda\mathbf{f}_2 = \mathbf{b}_2 \\ &\quad \mathbf{B}_2\mathbf{x}_2 + \mathbf{d}^2 = -\mathbf{t}_n - 2\mathbf{d}_n^1 \\ &\quad \mathbf{x}_2 \in \mathcal{K}_2 \end{aligned} \tag{9}$$

which give the optimal solutions $\mathbf{d}_n^1$ and $\mathbf{d}_n^2$. Each one of these problems has the structure of a second order cone program (SOCP), which can be solved by using standard optimisation packages. It can be verified that the sub-problems in (8) and (9) are in fact the implementations of the AAR algorithm for the sets $W(\lambda)$ and $Z(\lambda)$ defined in (6).

The stopping criteria in step 1.4. aims to detect, before an accurate value of α^k is computed, whether the AAR method is converging towards a fixed value (and α_n tends to zero), or diverging (β_n is increasing and α_n converges towards a non-zero value). In step 1.4.1. the value of λ^k is detected as a lower bound, while in 1.4.2. λ^k is detected as an upper bound. In Step 1.4.3 no identification can be said yet, and the iterative process ($n-$iterations) continues. The convergence of the algorithm is demonstrated in [RM] for general non-linear problems. We will next show its performance for some illustrative examples.

3 Results

Now we apply the AAR-based decomposition method for a rectangular domain depicted in Figure 4, and for three different number of elements. The number of elements considered (Nelem) are given in Table 1. The third column in the table also shows the accumulated number of iterations in the sub-problems, that is, the sum of the n_k sub-iterations for each master iteration k. The problem considers a fix left boundary, and the right boundary with an applied nominal traction equal to $\boldsymbol{g} = \{1, 1\}^T$. The domain has been partitioned horizontally, although it has been numerically tested that the results shown here do not depend on the actual partitioning and regularity of the mesh.

Problem	Nelem	$\sum_{k=1} n_k$
1	402	71
2	648	63
3	936	66

Table 1: Size and total number of sub-iterations of each problem

Problem 1					
$\lambda^* = 0.50280$					
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	n_k	$\Delta\lambda^{k-1}$
1	0.0	0.704068	0.352034	2	0.7041
2	0.352034	0.704068	**0.528051**	31	0.3520
3	0.352034	0.528051	0.440043	2	0.1760
4	0.440043	0.528051	0.484047	2	0.0880
5	0.484047	0.528051	**0.506049**	11	0.0440
6	0.484047	0.506049	0.495048	2	0.0220
7	0.495048	0.506049	0.500548	2	0.0110
8	0.495048	0.503299	**0.503299**	9	0.0083
9	0.501924	0.503299	0.501924	2	0.0014
10	0.502611	0.503299	0.502611	2	0.0007
11	0.502611	0.502955	**0.502955**	4	0.0003
12	0.502783	0.502955	0.502783	2	0.0002
–	0.502783	0.502955	$\lambda^* \approx 0.502869$	71	0.0002

Table 2: Numerical results of Problem 1

The optimal solution λ^* for each problem has been computed solving the global problem in (2) with SONIC solver [Lya04].

The numerical results of Problems 1-3 are reported in Table 2-4, respectively, where k indicates the number of master iterations, and n_k is the number of iterations taken by the sub-problem at each master iteration k. The second and third columns indicate the highest lower bound and the lowest upper bound at each master iteration, in such a way that $\lambda^* \in [\lambda_{lb}^{k-1}, \lambda_{ub}^{k-1}]$, and $\lambda^k = \frac{(\lambda_{lb}^{k-1} + \lambda_{ub}^{k-1})}{2}$. Numbers in bold font indicate that λ^k is an upper bound, and $\Delta\lambda^{k-1} = \lambda_{ub}^{k-1} - \lambda_{lb}^{k-1}$. In all our numerical tests, we have used the values $(tol, \epsilon_{1\alpha}, \epsilon_{2\alpha}) = (5E-4, 1E-4, 1E-4)$. It can be observed on the tables, and the plot in Figure 5, that whenever λ^k is an upper bound, the number of sub-iterations increases notoriously. Further work in detecting such upper bounds with less iterations is currently being undertaken.

4 Conclusions

We have presented and algorithm that solves the optimisation problem in limit analysis. The method explots the structure of the optimisation problem: a

Problem 2					
$\lambda^* = 0.50371$					
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	n_k	$\Delta\lambda^{k-1}$
1	0.0	0.745779	0.372889	2	0.7458
2	0.372889	0.745779	**0.559334**	21	0.3729
3	0.372889	0.559334	0.466112	2	0.1864
4	0.466112	0.559334	**0.512723**	9	0.0932
5	0.466112	0.512723	0.489417	2	0.0466
6	0.489417	0.512723	0.501070	2	0.0233
7	0.489417	0.506897	**0.506897**	7	0.0175
8	0.489417	0.503983	**0.503983**	9	0.0146
9	0.502527	0.503983	0.502527	2	0.0015
10	0.503255	0.503983	0.503255	2	0.0007
11	0.503619	0.503983	0.503619	2	0.0004
12	0.503619	0.503801	**0.503801**	3	0.0002
–	0.503619	0.503801	$\lambda^* \approx 0.503710$	63	0.0002

Table 3: Numerical results of Problem 2

Problem 3					
$\lambda^* = 0.50507$					
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	n_k	$\Delta\lambda^{k-1}$
1	0.0	0.705498	0.352749	2	0.7055
2	0.352749	0.705498	**0.529123**	29	0.3527
3	0.352749	0.529123	0.440936	2	0.1764
4	0.440936	0.529123	0.485030	2	0.0882
5	0.485030	0.529123	**0.507077**	11	0.0441
6	0.485030	0.507077	0.496053	2	0.0220
7	0.496053	0.507077	0.501565	2	0.0110
8	0.501565	0.507077	0.504321	2	0.0055
9	0.504321	0.507077	**0.505699**	7	0.0028
10	0.504321	0.505699	0.505010	2	0.0014
11	0.505010	0.505699	**0.505354**	3	0.0007
12	0.505010	0.505354	**0.505182**	2	0.0003
–	0.505010	0.505182	$\lambda^* \approx 0.505096$	66	0.0002

Table 4: Numerical results of Problem 3

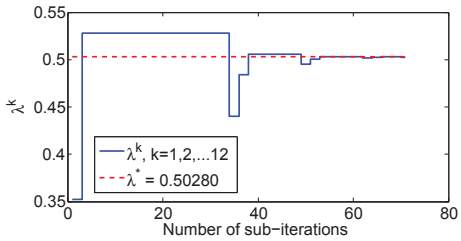

Figure 5: Evolution of the load factor for each sub-iteration in Problem 1.

single scalar on the objective function.

Assuming that the cost of the optimisation problem increases approximately with in the order of $nelem^2$, the total number of iterations obtained of the decomposed algorithm is still not competitive, although the number of iterations that other standard procedures would require, such as Bendres or dual decomposition, has been recuded by one order of magnitude [RM]. However, we note that the number of sub-iterations does not scale with the problem size, and that the memory requirements have been reduced when solving the decomposed algorithm.

The reduction of the number of iterations when the estimate of the load factor λ^k is an upper bound is under investigation. One possible venue would be to update the value of λ^{k+1} with a value closer to λ^k_{lb}, instead of the average computed in Step 3.2. In this manner, more lower bounds would be in general detected, which require at most 4 iterations in the examples shown.

5 Acknowledgements

The authors acknowledge the financial support of the Spanish Ministry of Economy and Competitiveness, through the research grant Nr. DPI2013-43727-R.

References

[BC10] H H Bauschke and P L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. CMS Books in Mathematics. Springer, 2010.

[Bd14] J Bleyer and P de Buhan. A computational homogenization approach for the yield design of periodic thin plates. Part I: Construction of the macroscopic strength criterion. *Int. J. Solids Struct.*, 51(13):2448–2459, 2014.

[Ben62] J F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4:238–252, 1962.

[BXMM07] S Boyd, L Xiao, A Mutapcic, and J Mattingley. Notes on decomposition methods. Technical report, Stanford University, 2007. Notes of EE364B course.

[CCMGB06] AJ Conejo, E Castillo, R Mínguez, and R García-Bertrand. *Decomposition Techniques in Mathematical Programming*. Springer, The Netherlands, 2006.

[CT94] G Chen and M Teboulle. A proximal-based decomposition method for convex minimization problems. *Math. Progr. Ser. A*, 64:81–101, 1994.

[DW60] G B Dantzig and Wolfe. Decomposition principle for linear programs. *Oper. Res.*, 8(1):101–111, 1960.

[Geo72] A M Geoffrion. Generalized Benders Decomposition. *J. Optim. Th. Appl.*, 10(4):238–252, 1972.

[GSP10] M Gilbert, CC Smith, and TJ Pritchard. Masonry arch analysis using discontinuity layout optimisation. *Proc. Inst. Civil Eng. - Eng. Comp. Mech.*, 163(3):155–166, 2010.

[Kan83] I Kaneko. A decomposition procedure for large-scale optimum plastic design problems. *Int. J. Num. Meth. Engng.*, 19:873–889, 1983.

[KLS07] K Krabbenhøft, A V Lyamin, and S W Sloan. Formulation and solution of some plasticity problems as conic programs. *Int. J. Solids Struct.*, 44:1533–1549, 2007.

[KPSP10] Z Kammoun, F Pastor, H Smaoui, and J Pastor. Large static problem in numerical limit analysis: A decomposition approach. *Int. J. Num. Anal. Meth. Geomech.*, 34:1960–1980, 2010.

[LS02] A V Lyamin and S W Sloan. Lower bound limit analysis using non-linear programming. *Int. J. Num. Meth. Engng.*, 55:576–611, 2002.

[LSKH05] A V Lyamin, S W Sloan, K Krabbenhøft, and M Hjiaj. Lower bound limit analysis with adaptive remeshing. *Int. J. Num. Meth. Engng.*, 63:1961–1974, 2005.

[Lya04] A V Lyamin. Sonic. Solver for second order conic programing. 2004.

[MBHP09] J J Muñoz, J Bonet, A Huerta, and J Peraire. Upper and lower bounds in limit analysis: adaptive meshing strategies and discontinuous loading. *Int. J. Num. Meth. Engng.*, 77:471–501, 2009.

[MLH13] J J Muñoz, A Lyamin, and A Huerta. Stability of anchored sheet wall in cohesive-frictional soils by FE limit analysis. *Int. J. Num. Anal. Meth. Geomech.*, 37(9):1213–1230, 2013.

[MOS05] MOSEK ApS. The MOSEK optimization tools version 3.2 (Revision 8). *User's Manual and Reference*, 2005. Avail. http://www.mosek.com.

[MRLH13] J J Muñoz, N Rabiei, A Lyamin, and A Huerta. *Direct Methods for Limit States in Structures and Materials*, chapter Computation of bounds for anchor problems in limit analysis and decomposition techniques, pages 79–100. Springer Verlag, 2013.

[MS10] RS Merifield and CC Smith. The ultimate uplift capacity of multi-plate strip anchors in undrained clay. *Comp. Geotech.*, 37(4):504–514, 2010.

[PLP09] F Pastor, E Loute, and J Pastor. Limit analysis and convex programming: A decomposition approach of the kinematic mixed method. *Int. J. Num. Meth. Engng.*, 78:254–274, 2009.

[RM] N Rabiei and J J Muñoz. AAR-based decomposition method for non-linear convex optimization. Submitted for publication.

[TTT06] K C Toh, M J Todd, and R H R H Tütüncü. On the implementation and usage of SDPT3 : a Matlab software package for semidefinite-quadratic-linear programming, version 4.0. Technical report, National University of Singapore, July 2006.